ELSEVIER

Theoretical Computer Science 292 (2003) 525–546

Theoretical Computer Science

www.elsevier.com/locate/tcs

Discovering characteristic expressions in literary works

Masayuki Takeda[a,b,*], Tetsuya Matsumoto[a,1], Tomoko Fukuda[c], Ichiro Nanri[c]

[a] *Department of Informatics, Kyushu University 33, Fukuoka 812-8581, Japan*
[b] *PRESTO, Japan Science and Technology Corporation (JST), Kawaguchi 332-0012, Japan*
[c] *Junshin Women's Junior College, Fukuoka 815-0036, Japan*

Abstract

We attempt to extract characteristic expressions from literary works. That is, given two collections of literary works, one of which is written by a particular author (positive examples) and the other by a different author (negative examples), the problem is to find expressions that appear frequently in the positive examples but which are seldom found in the negative examples. This is considered as a special case of *the optimal pattern discovery* from textual data, in which only *the substring patterns* are considered. One approach would be to create a list of text substrings sorted according to *goodness*, and to scrutinize the first part of the list by human efforts. Since there is no word boundary in Japanese texts, a substring is often a fragment of a word or phrase. A method to assist domain experts who are involved in this task is a key problem. In this paper, we propose partitioning the text substrings into equivalence classes under an equivalence relation on strings, originally defined by Blumer et al. (J. ACM 34(3) (1987) 578). The equivalence relation has the desirable property that all members of each equivalence class necessarily have a unique goodness value. This idea effectively reduces the inefficiency of the task of evaluating mined patterns. We also present a method for browsing possible superstrings of a focused string as well as its context. We report successful results with two pairs of anthologies of classical Japanese poems. We expect that the extracted expressions may lead to discovering overlooked aspects of individual poets. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Optimal pattern discovery; Substring statistics; Waka poetry; Characteristic expressions; Literary works

* Corresponding author. Department of Informatics, Kyushu University, 33, Fukuoka 812-8581, Japan.
E-mail addresses: takeda@i.kyushu-u.ac.jp (Masayuki Takeda), tomoko-f@muc.biglobe.ne.jp (Tomoko Fukuda), nanri-i@msj.biglobe.ne.jp (Ichiro Nanri).
[1] Presently at NTT DoCoMo, Inc.

0304-3975/03/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0304-3975(02)00185-8

1. Introduction

Analysis of expression is one of the most fundamental methods in literary studies. Word statistics are useful for this. Literary researchers are often interested in:
(1) high-frequency words,
(2) low-frequency words, and
(3) words that distinguish between two given sets of texts, i.e., words that frequently appear in one set, but do not do so in the other.
In this paper we focus on the problems concerning the third item above.

If we wish to use word statistics for analyzing Japanese literary works, we need to perform word segmentation, because there is no word boundary in Japanese texts. This is a difficult and time-consuming task. For example, it is reported in [12] that it took 8 years to build a part-of-speech-tagged corpus of "the Tale of Genji". In the case of Waka poetry, building such corpora is much more difficult because of the frequently used poetic device, called "Kake-kotoba",[2] which exploits the ambiguities of a word or part of a word. A unique word segmentation is essentially impossible in such a situation.

Recently, some researchers noticed that using substring statistics instead of word statistics is useful in expression analysis, although a substring could be a fragment of a word or phrase. For example, Kondo [9] proposed an expression analysis method based on n-gram statistics.[3] She extracted the strings that differentiate the poems by male authors from those by female authors in Kokin-Shū, the most famous imperial anthology. She examined these and found some gender-specific expressions. The n-gram statistics are essentially the same as substring statistics since n cannot be fixed. This work opened the door for the application of substring statistics to Japanese literary works.

To ease her burden, Kondo restricted herself to the substrings which:
- are of length from three to seven,
- occur more than once, and
- are used only by male writers (not used by female writers).

We will show that such a restriction can be removed by exploiting the combinatorial properties on strings.

In this paper, we address the problem of discovering characteristic expressions from anthologies of classical Japanese poems. We believe that the support of domain experts involved in the interpretation/evaluation of mined patterns is a key to success, and propose:
- partitioning text substrings into equivalence classes under an equivalence relation on strings, so that domain experts can examine the equivalence classes one by one; and
- to show left and right contexts of a string as trees.

[2] A homonymic punning where the double meaning of a word or part of a word is exploited. In English it is usually called *the pivot words* because it is used as a pivot between two series of sounds with overlapping syntactical and semantic patterns.

[3] n-Gram statistics are frequency comparisons of text substrings of equal length n.

あ は と み る	A-HA-TO-MI-RU	(5 syllables)
あ は ち の し ま の	A-HA-CHI-NO-SHI-MA-NO	(7 syllables)
あ は れ さ へ	A-HA-RE-SA-HE	(5 syllables)
の こ る く ま な く	NO-KO-RU-KU-MA-NA-KU	(7 syllables)
す め る よ の つ き	SU-ME-RU-YO-NO-TSU-KI	(7 syllables)

Fig. 1. An example of a Waka poem. It has five lines and 31 syllables. One syllable is written as one kana character on the left, and is romanized on the right, where hyphens were inserted between syllables as boundaries.

Using the suggested method we extracted the strings that differentiate two pairs of anthologies, academically scrutinized them, and succeeded in finding the characteristic expressions. We compared Sanka-Shū by Saigyō with Shūgyoku-Shū by Jien, and Shūi-Gusō by Fujiwara-no-Teika with Tameie-Shū by his son, Fujiwara-no-Tameie separately, and procured expressions highlighting the differences between each of the two anthologies.

Saigyō and Jien were both priests, but their lives were a considerable contrast in status and social circumstances. Being a son of the famous regent, Jien could not break with the government. However, it is well known that Jien sought help from Saigyō in his final days, not only about poetic composition, but also about the way of life. For example, he confessed his plan to become a hermit.

On the other hand, Tameie was rigorously trained by his own father, Teika, because he was a direct descendant of the Mikohidari dynasty famous for producing poets. Tameie referred to many of Teika's poems.

In each pair of anthologies there necessarily exist similar poems. However, as we have demonstrated, we could collect certain differences in expressions. This, we expect, will possibly lead to discoveries about overlooked aspects of the individual poets.

This work is a multidisciplinary study of literature and computer science. The second-last author is a Waka researcher, and the final author is a linguist in the Japanese language.

Here, we explain Waka poetry. A Waka poem has five lines and 31 syllables, arranged five-seven-five-seven-seven. See Fig. 1 for an example of a Waka poem. In this figure, the poem is written in kana characters as well as in romanized form, and we note that one kana character corresponds to one syllable in Japanese. In the remainder of this paper, we will often quote Waka poems in romanized form, where hyphens are inserted between syllables as boundaries.

2. Background

In classical Japanese poetry, Waka, there are strict rules about the choice and combination of poetic words. For instance, the word "Uguisu" (Japanese bush warbler) should be used linked with the word "Ume" (plum-blossom). It is significant in Waka

studies, therefore, to consider the way poets learned such rules and developed their own expressions. We are interested in investigating the way they learned certain expressions from their predecessors.

From this point of view, we have established a method for semi-automatically extracting from a Waka database, poems similar in expression [14]. Using this method, we discovered previously unnoticed affinities of some poems with earlier poems. This raised an interesting issue for Waka studies, to which we were able to give a convincing conclusion [7,6].

While continuing to find affinities between Waka poems, it is also necessary to place some additional conditions on the method when we compare poets in a parent–child or teacher–student relationship. It is easily inferred that poet A (the master poet) greatly influences poet B (the disciple poet), and that the poems by poet B may have frequently alluded to those by poet A. In fact, many scholars have already noticed such apparent literary relationships. In such cases, it is much more important to clarify the differences than to enumerate the affinities. For example, when poet B hardly ever adopts the expressions frequently used by poet A, it will enable us to study their relationship in a different way. In this paper, we will compare two poets' private anthologies and try to make clear the differences and features in the similar expressions on the basis of their frequencies. This is derived from the same methodology we have been practicing so far.

3. Method

3.1. Preliminaries

Let Σ be a finite alphabet. An element of Σ^* is called a *string*. Strings x, y, and z are said to be a *prefix*, *substring*, and *suffix* of the string $u = xyz$, respectively. A string u is said to be a *superstring* of a string y if y is a substring of u. The length of a string u is denoted by $|u|$. The empty string is denoted by ε, that is, $|\varepsilon| = 0$. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. The ith symbol of a string u is denoted by $u[i]$ for $1 \leqslant i \leqslant |u|$, and the substring of a string u that begins at position i and ends at position j is denoted by $u[i:j]$ for $1 \leqslant i \leqslant j \leqslant |u|$. For convenience, let $u[i:j] = \varepsilon$ for $j < i$. Let $Sub(w)$ denote the set of substrings of w, and let $Sub(S) = \bigcup_{w \in S} Sub(w)$ for a set S of strings. The elements of $Sub(S)$ are called *substrings of* S. For a set S of strings, let $\|S\|$ denote the total length of the strings in S, and let $|S|$ denote the cardinality of S. Let w^R denote the reversed string of a string w, and let $S^R = \{w^R \mid w \in S\}$ for a set S of strings.

3.2. Optimal pattern discovery

Shimozono et al. [13] formulated the problem of text data mining as an instance of *optimal pattern discovery* [11]. Let Π be a class of *patterns*. Assume that each pattern π in Π is associated with a language $L(\pi) \subseteq \Sigma^*$. Let $\psi : [0,1] \to \boldsymbol{R}$ be a symmetric, concave, real-valued function called the *impurity function* [4] to quantify the skewness of the class distribution after the classification by a pattern in Π. The *classification*

error $\psi_1(r) = \min(r, 1-r)$, the *information entropy* $\psi_2(r) = -r \log r - (1-r)\log(1-r)$, and the *Gini index* $\psi_3(r) = 2r(1-r)$ are examples of ψ [4].

Given two disjoint subsets *Pos* and *Neg* of Σ^+, a pattern discovery algorithm tries to find a pattern π in Π that optimizes the *evaluation function*

$$G^{\psi}_{Pos,Neg}(\pi) = (p_1 + n_1)\psi\left(\frac{p_1}{p_1 + n_1}\right) + (p_0 + n_0)\psi\left(\frac{p_0}{p_0 + n_0}\right),$$

where $p_1 = |L(\pi) \cap Pos|$ and $n_1 = |L(\pi) \cap Neg|$, and $p_0 = |Pos| - p_1$ and $n_0 = |Neg| - n_1$. The problem is:

Definition 1 (Optimal pattern discovery with respect to ψ). Given two disjoint subsets *Pos* and *Neg* of Σ^+, find a pattern π in Π that minimizes the cost $G^{\psi}_{Pos,Neg}(\pi)$.

A *k-proximity d-phrase association pattern* (a (k,d)-proximity pattern) is a pair of a sequence $p_1, \ldots, p_d$ of phrases in Σ^+ and a bounded gap length k, called *proximity*. It matches a string w if there exists a sequence $j_1, \ldots, j_d$ of integers such that each phrase p_i occurs at position j_i of w, and $0 < j_{i+1} - j_i \leqslant k$ for each $i = 1, \ldots, d-1$.[4] Shimozono et al. [13] presented an efficient algorithm for the class of (k,d)-proximity patterns. Although only the classification error measure is dealt with in [13], the algorithm works for many other measures [1].

Our problem is regarded as a special case of this problem, where $d = 1$. That is, we deal with only the patterns of the form $\bigstar w \bigstar$, where w is an arbitrary non-empty string, and $\bigstar$ is a wildcard that matches any string. We call such patterns the *substring patterns*. We have a trivial $O(m)$ time and space algorithm since there are essentially $O(m)$ candidates for the optimal substring w, where m is the total length of strings in $S = Pos \cup Neg$. Therefore, it appears the main difficulty is to find an appropriate statistical measure.

However, we found that a considerable number of the mined patterns were obvious and worthless, although we tested several statistical measure functions for our problem. In practice it seems that no matter what measure is used, we cannot remove these worthless patterns from the results. Discovery requires an effort by domain experts to examine the first part of a list of patterns arranged in decreasing order of goodness. This corresponds to the step of *interpreting mined patterns*, a "postprocessing" of data mining in the knowledge discovery process [5]. We believe that this step to support the domain experts is a key to success. We tackle this problem with the weapon of *stringology* [8].

3.3. Inefficiency in domain experts' tasks

We take the following approach to our problem:
(1) Choose an appropriate statistical measure G.

[4] A (k,d)-proximity pattern is closely related to the pattern $\bigstar p_1 \bigstar \cdots \bigstar p_d \bigstar$ in which a gap symbol $\bigstar$ is limited to match only a string of, at most, length k. However, these are not identical. Note that the phrases p_i in a (k,d)-proximity pattern may have overlaps because $j_i + |p_i|$ can be greater than j_{i+1}.

Table 1
A partial list of substrings of poems in two imperial anthologies Kokin-Shū and Gosen-Shū. We excluded substrings that cover the boundaries between the lines of a poem. The substrings with the same goodness value were sorted lexicographically with respect to the order of kana characters, although they are romanized here

Rank	G	P_1	N_1	String
⋮				
234	0.6836	11	33	TSU-YU-NO
235	0.6836	4	0	A-YA-MA-TA-RE
236	0.6836	4	0	A-YA-MA-TA-RE-KE
237	0.6836	4	0	KA-KE-KA
⋮				
303	0.6836	4	0	FU-RI-TA
304	0.6836	4	0	HO-NI-I-TE-TE
305	0.6836	4	0	MA-TA-RE-KE
306	0.6836	4	0	MI-YA-KI
307	0.6836	4	0	MU-NO
308	0.6836	4	0	MO-KA-MO
309	0.6836	4	0	MO-TO-YU
310	0.6836	4	0	MO-RU-TO
311	0.6836	4	0	YA-CHI-YO
312	0.6836	4	0	YA-MA-TA-RE
313	0.6836	4	0	YA-MA-TA-RE-KE
314	0.6836	4	0	RA-ME-TO
315	0.6836	4	0	RU-TO-I
⋮				

(2) Create a list of substrings of the strings in $S = Pos \cup Neg$, sorted according to the values of G.
(3) Have the upper part of the list evaluated by domain experts.

When two or more strings have the same value in Step 2, it is reasonable to arrange them in lexicographical order. Table 1 is a partial list of substrings for Kokin-Shū and Gosen-Shū, the first and second imperial anthologies. It is only by coincidence that the strings "A-YA-MA-TA-RE" (ranked 235th) and "KA-KE-KA" (ranked 237th) have the same frequencies. However, the situation is different for the strings "A-YA-MA-TA-RE" and "A-YA-MA-TA-RE-KE" (ranked 235th and 236th). The former is a prefix of the latter, and every time the former appears, it is followed by a character "KE". In this case, every occurrence of the strings "MA-TA-RE-KE", "YA-MA-TA-RE", and "YA-MA-TA-RE-KE" (ranked 305th, 312th, and 313th, respectively) was a substring of the string "A-YA-MA-TA-RE-KE", and therefore all of these strings have the same goodness value. It is desirable that strings in such a relationship appear in one section of the list. A simple lexicographical sorting, however, disperses them as in Table 1. This causes inefficiency in the task of evaluating strings in the list.

3.4. Reducing inefficiency

To reduce this inefficiency, we partition the substrings of $S = Pos \cup Neg$ into equivalence classes by using an equivalence relation on Σ^*, denoted by $\equiv$, which was first defined by Blumer et al. [2], and has the following properties:

- Each equivalence class has a unique longest string that contains any other member as a substring, and we regard it as the *representative* of the class.
- All the strings in an equivalence class have the same frequencies in *Pos* and in *Neg*, respectively, and therefore have the same value of goodness.
- The number of equivalence classes is linear with respect to the total length of strings in $S = Pos \cup Neg$.

The basic idea is to create a list of equivalence classes, not just a list of substrings. In this list we can sort the equivalence classes according to the goodness values since all members of each equivalence class have the same goodness value. See Table 2.

It should be stated that the data structure called the *suffix tree* [3] exploits a similar and more popular equivalence relation, denoted by $\equiv_{\mathrm{L}}$, which also satisfies the above three conditions. The equivalence relation $\equiv_{\mathrm{L}}$ is a refinement of $\equiv$, and therefore includes many more equivalence classes. See Table 4 for a comparison. From the viewpoint of computational complexity, this is not a significant difference because the two equivalence relations both satisfy the third condition above. However, the difference is crucial for researchers who must check the candidate strings. Table 3 shows the numbers of non-degenerate equivalence classes for the strings of Kokin-Shū and Gosen-Shū, under the three equivalence relations $\equiv_{\mathrm{L}}$, $\equiv_{\mathrm{R}}$, and $\equiv$, formally defined in the next section. From this table, it is observed that the number of equivalence classes under $\equiv_{\mathrm{L}}$ is approximately four times that under $\equiv$.

Although each of the equivalence classes in $\equiv$ may contain $O(m^2)$ strings, where m is the length of its representative, a compact representation is possible in which only the representative and the minimal members are shown. The number of minimal members is at most m, as proved in Section 5.4.

4. Equivalence relations on strings

In this section, we give formal definitions of the equivalence relations of strings, and present some of their properties.

4.1. Definition

Definition 2. Let S be a non-empty finite subset of Σ^+. For any x in $Sub(S)$, let

$$Beginpos_S(x) = \{\langle w, j\rangle \mid w \in S,\ 0 \leqslant j \leqslant |w|,\ x = w[j+1 : j+|x|]\},$$
$$Endpos_S(x) = \{\langle w, j\rangle \mid w \in S,\ 0 \leqslant j \leqslant |w|, x = w[j-|x|+1 : j]\}.$$

For any $x \notin Sub(S)$, let $Beginpos_S(x) = Endpos_S(x) = \emptyset$.

Table 2
A list of equivalence classes for Kokin-Shū and Gosen-Shū. The symbol • indicates the representative of each equivalence class. When two or more classes have the same goodness value, we arranged them in lexicographical order of their representatives

Rank	G	P_1	N_1	Equivalence class
				⋮
187	0.6836	11	33	• TSU-YU-NO
188	0.6836	4	0	• A-YA-MA-TA-RE-KE
				YA-MA-TA-RE-KE
				MA-TA-RE-KE
				A-YA-MA-TA-RE
				YA-MA-TA-RE
189	0.6836	4	0	• KA-KE-KA
190	0.6836	4	0	• KA-SHI-RA
				⋮
197	0.6836	4	0	• KO-HI-SHI-KA-RI-KE-RU
198	0.6836	4	0	• KO-HI-SHI-KA-RU-HE
				HI-SHI-KA-RU-HE
199	0.6836	4	0	• KO-HI-SHI-KI-TO-KI
				HI-SHI-KI-TO-KI
				SHI-KI-TO-KI
				KO-HI-SHI-KI-TO
				HI-SHI-KI-TO
200	0.6836	4	0	• KO-HI-YA-WA-TA-RA-MU
				HI-YA-WA-TA-RA-MU
				KO-HI-YA-WA-TA-RA
				HI-YA-WA-TA-RA
				KO-HI-YA-WA-TA
				HI-YA-WA-TA
				KO-HI-YA-WA
				HI-YA-WA
201	0.6836	4	0	• SA-KI-SO
				⋮

Table 3
Comparison of the numbers of non-degenerate equivalence classes for Kokin-Shū and Gosen-Shū for the three equivalence relations, $\equiv_L$, $\equiv_R$, and $\equiv$. We show two cases: (A) S is the set of poems, and (B) S is the set of lines of poems

	$\lvert S\rvert$	$\lVert S\rVert$	$\lvert Sub(S)\rvert$	No. of non-degen. equiv. classes		
				$\equiv_L$	$\equiv_R$	$\equiv$
A	2518	78,727	1,006,325	92,977	92,897	23,239
B	9515	61,397	79,963	40,049	39,954	20,448

Table 4
Comparison of two equivalence relations. The anthologies used were Kokin-Shū and Gosen-Shū. The left and the right lists are based on the equivalence relations $\equiv_L$ and $\equiv$, respectively. Here, we showed only the equivalence classes with $P_1 = 5$ and $N_1 = 0$, where P_1 and N_1 are the frequencies in Kokin-Shū and Gosen-Shū, respectively. The numbers of such classes under $\equiv_L$ and $\equiv$ are, respectively, 37 (ranked 60th–96th) and 29 (ranked 59th–87th). Two equivalence classes, ranked 79th and 82nd on the left list, are merged into one equivalence class, ranked 78th on the right. Three equivalence classes, ranked 65th, 80th, and 95th on the left, are merged into one equivalence class, ranked 63rd on the right. Thus, $\equiv_L$ is a refinement of $\equiv$

Rank	Equivalence class under $\equiv_L$	Rank	Equivalence class under $\equiv$
60	• A-NA-U	59	• A-NA-U
	⋮		⋮
64	• KU-SHI-KA	62	• KA-YO-HI-CHI
65	• KU-RE-NA-WI-NO		YO-HI-CHI
66	• KU-HI-YA	63	• KU-RE-NA-WI-NO
	⋮		RE-NA-WI-NO
			NA-WI-NO
78	• NA-KU-SHI-KA	64	• KO-HI-YA
79	• NA-SO-CHI-RI-KE-RU		⋮
	NA-SO-CHI-RI-KE		
	NA-SO-CHI-RI	76	• NA-KU-SHI-KA
	NA-SO-CHI		KU-SHI-KA
80	• NA-WI-NO	77	• NU-NO
81	• NU-NO	78	• HA-NA-SO-CHI-RI-KE-RU
82	• HA-NA-SO-CHI-RI-KE-RU		NA-SO-CHI-RI-KE-RU
	HA-NA-SO-CHI-RI-KE		HA-NA-SO-CHI-RI-KE
	HA-NA-SO-CHI-RI		NA-SO-CHI-RI-KE
	HA-NA-SO-CHI		HA-NA-SO-CHI-RI
83	• HA-MA-SHI-MO		NA-SO-CHI-RI
	⋮		HA-NA-SO-CHI
			NA-SO-CHI
94	• RU-RA-ME-YA	79	• HA-MA-SHI-MO-NO-WO
95	• RE-NA-WI-NO		HA-MA-SHI-MO-NO
96	• WO-TO-TO		⋮
		87	• WO-TO-TO

For example, if $S = \{babbc, ababb\}$, then $Beginpos_S(a) = Beginpos_S(ab) = \{\langle babbc, 1\rangle, \langle ababb, 0\rangle, \langle ababb, 2\rangle\}$, $Beginpos_S(c) = \{\langle babbc, 4\rangle\}$, and $Endpos_S(bb) = Endpos_S(abb) = Endpos_S(babb) = \{\langle babbc, 4\rangle, \langle ababb, 5\rangle\}$.

For the remainder of this paper we omit the set S, and write simply *Beginpos* and *Endpos*.

Definition 3. Let x and y be arbitrary strings in Σ^*. We write $x \equiv_L y$ if $Beginpos(x) = Beginpos(y)$, and write $x \equiv_R y$ if $Endpos(x) = Endpos(y)$. The equivalence class of a string $x \in \Sigma^*$ with respect to $\equiv_L$ (resp. $\equiv_R$) is denoted by $[x]_{\equiv_L}$ (resp. $[x]_{\equiv_R}$).

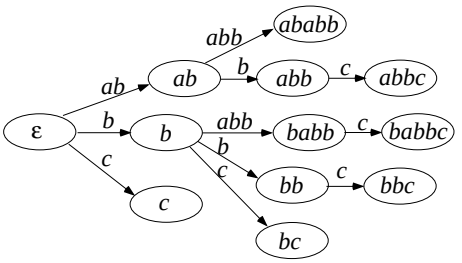

Fig. 2. Suffix tree for $S = \{babbc, ababb\}$.

If $S = \{babbc, ababb\}$, then $[\varepsilon]_{\equiv_L} = [\varepsilon]_{\equiv_R} = \{\varepsilon\}$, $[a]_{\equiv_L} = \{a, ab\}$, $[bb]_{\equiv_R} = \{bb, abb, babb\}$, and $[c]_{\equiv_R} = \{c, bc, bbc, abbc, babbc\}$.

Note that all strings that are not in $Sub(S)$ form one equivalence class under $\equiv_L$ ($\equiv_R$). This equivalence class is called the *degenerate* class. All other classes are called *non-degenerate*.

It follows from the definition of $\equiv_L$ that, if x and y are strings in the same non-degenerate class under $\equiv_L$, then either x is a suffix of y, or vice versa. Therefore, each non-degenerate equivalence class in $\equiv_L$ has a unique longest member. Similar discussion holds for $\equiv_R$.

Definition 4. For any string x in $Sub(S)$, let $\overrightarrow{x}$ and $\overleftarrow{x}$ denote the unique longest members of $[x]_{\equiv_L}$ and $[x]_{\equiv_R}$, respectively.

For any string x in $Sub(S)$, there exists a unique pair of strings α and β such that $\overleftarrow{x} = \alpha x$ and $\overrightarrow{x} = x\beta$. In the running example, we have $\overleftarrow{\varepsilon} = \overrightarrow{\varepsilon} = \varepsilon$, $\overrightarrow{a} = ab$, $\overleftarrow{b} = b$, $\overleftarrow{bb} = babb$, $\overrightarrow{bab} = babb$, and $\overleftarrow{c} = babbc$.

The suffix tree and the prefix tree for S are useful to illustrate the operators $\overleftarrow{(\cdot)}$ and $\overrightarrow{(\cdot)}$. The strings $x \in Sub(S)$ with $x = \overrightarrow{x}$ are the nodes of the suffix tree of a set S of strings, and the other strings are not explicitly represented. Fig. 2 shows the suffix tree for $S = \{babbc, ababb\}$. For example, $\overrightarrow{aba} = ababb$ implies that an edge traversal from the root ε, navigated by the string aba, ends on the edge to the node $ababb$. Symmetrically, the prefix tree for S is a data structure in which only the strings $x \in Sub(S)$ with $x = \overleftarrow{x}$ are represented explicitly as nodes. Fig. 3 shows the prefix tree for $S = \{babbc, ababb\}$, which is identical to the suffix tree for $S^R = \{cbbab, bbaba\}$. For example, $\overleftarrow{abb} = babb$ means that an edge traversal from the root, navigated by the string abb reading in the reverse direction, ends on the edge to the node $babb$. On the other hand, Fig. 4 shows the directed acyclic word graph (DAWG for brevity) [3] for S. Note that the nodes are the non-degenerate equivalence classes in $\equiv_R$. The DAWG is the smallest automaton that recognizes the set of suffixes of the strings of S if we appropriately designate some nodes as *final states*.

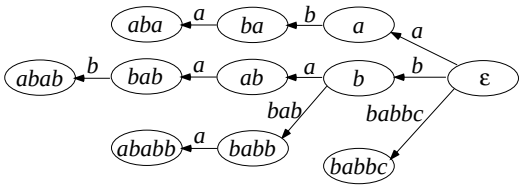

Fig. 3. Prefix tree for $S = \{babbc, ababb\}$. This is the same as the suffix tree for $S^{\mathrm{R}} = \{cbbab, bbaba\}$.

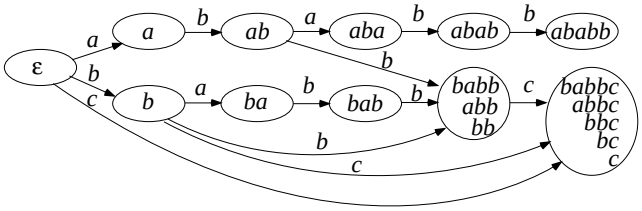

Fig. 4. DAWG for $S = \{babbc, ababb\}$.

Definition 5. For any string x in $Sub(S)$, let $\overleftrightarrow{x}$ be the string $\alpha x \beta$ such that α and β are the strings satisfying $\overleftarrow{x} = \alpha x$ and $\overrightarrow{x} = x\beta$.

In the running example, $\overleftrightarrow{\varepsilon} = \varepsilon$, $\overleftrightarrow{a} = ab$, $\overleftrightarrow{b} = b$, $\overleftrightarrow{ab} = ab$, $\overleftrightarrow{abb} = babb$, $\overleftrightarrow{bab} = babb$, and $\overleftrightarrow{c} = babbc$.

Definition 6. Strings x and y are said to be *equivalent* on S if and only if:
(1) $x \notin Sub(S)$ and $y \notin Sub(S)$, or
(2) $x, y \in Sub(S)$ and $\overleftrightarrow{x} = \overleftrightarrow{y}$.

This equivalence relation is denoted by $x \equiv y$. The equivalence class of x under $\equiv$ is denoted by $[x]_{\equiv}$.

Note that, for any string x in $Sub(S)$, the string $\overleftrightarrow{x}$ is the longest member of $[x]_{\equiv}$. We regard the string $\overleftrightarrow{x}$ as the *representative* of the non-degenerate equivalence class $[x]_{\equiv}$. Intuitively, $\overleftrightarrow{x} = \alpha x \beta$ means that:

- Every time x occurs in S, it is preceded by α and followed by β.
- Strings α and β are as long as possible.

Now, we are ready to define the prime substrings.

Definition 7. A string x in $Sub(S)$ is said to be *prime* if $\overleftrightarrow{x} = x$.

4.2. Properties of equivalence relations

Lemma 1 (Blumer et al. [2]). *The equivalence relation $\equiv$ is the transitive closure of the relation $\equiv_{\mathrm{R}} \cup \equiv_{\mathrm{L}}$.*

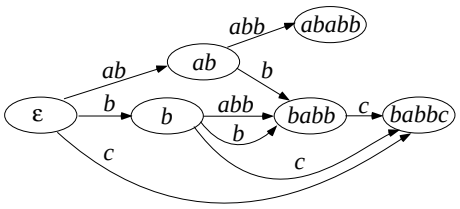

Fig. 5. Compact DAWG for $S=\{babbc, ababb\}$. The edge labeled by $\sigma\gamma$ from the node x to the node y means that $x\sigma$ is equivalent to y under $\equiv$, where $x, y \in Prime(S)$, $\sigma \in \Sigma$, $\gamma \in \Sigma^*$. For instance, the arrow labeled "*abb*" from the node "*b*" to the node "*babb*" means that *ba* is equivalent to *babb* under $\equiv$.

It follows from the above lemma that $\overleftrightarrow{x} = (\overrightarrow{x}) = (\overleftarrow{x})$ for any string x in $Sub(S)$.

Recall that the number of all substrings of S is $O(\|S\|^2)$. This can be reduced to $O(\|S\|)$ by considering only the substrings x such that $\overrightarrow{x} = x$. In fact, the suffix tree is a data structure that exploits this property. Similarly, the DAWG achieves its $O(\|S\|)$ space complexity by identifying every substring x with the substring $\overleftarrow{x}$. The number of prime substrings of S is also $O(\|S\|)$. The next lemma gives tight bounds.

Lemma 2 (Blumer et al. [2]). *Assume* $\|S\| > 1$. *The number of non-degenerate equivalence classes in* $\equiv_L$ ($\equiv_R$) *is at most* $2\|S\| - 1$. *The number of non-degenerate equivalence classes in* $\equiv$ *is at most* $\|S\| + |S|$.

Let $Prime(S)$ be the set of prime substrings of S, i.e., $Prime(S) = \{\overleftrightarrow{x} \mid x \in Sub(S)\}$.

Definition 8. The symmetric compact DAWG for S is the triple (V, E_L, E_R), where $V = Prime(S)$ is the set of vertices, and $E_L, E_R \subseteq V \times V \times \Sigma^+$ are two types of labeled edges defined by:

$$E_L = \{(x, \gamma\sigma x\delta, \gamma\sigma) \mid x \in V, \sigma \in \Sigma, \gamma, \delta \in \Sigma^*, \gamma\sigma x\delta = \overleftrightarrow{\sigma x}\},$$
$$E_R = \{(x, \delta x\sigma\gamma, \sigma\gamma) \mid x \in V, \sigma \in \Sigma, \delta, \gamma \in \Sigma^*, \delta x\sigma\gamma = \overleftrightarrow{x\sigma}\}.$$

The compact DAWG for S is the directed acyclic graph specified by (V, E_R).

Fig. 5 shows the compact DAWG for the running example. Compared with the suffix tree of Fig. 2 and with the DAWG of Fig. 4, the nodes represent only the prime substrings, and therefore it is expected that the number of nodes is smaller than those of the suffix tree and the DAWG.

Fig. 6 shows the symmetric compact DAWG for the running example.

Lemma 3 (Blumer et al. [2]). Both the compact DAWG and the symmetric compact DAWG for S can be constructed in $O(\|S\|)$ time and space.

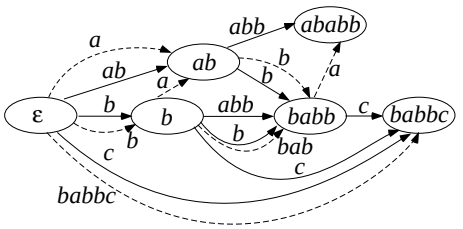

Fig. 6. Symmetric compact DAWG for $S = \{babbc, ababb\}$. The solid and the broken arrows represent the edges in E_R and E_L, respectively.

5. Superstring–substring relationship

5.1. Why this relation is required

In our scenario, domain experts scrutinize the equivalence classes singly in order to find an equivalence class that contains an 'interesting' substring. For evaluation/interpretation of a string, domain experts always need to access the context of each appearance. That is, they have to know what string immediately precedes (follows) each occurrence of the same string. This is essentially the same as seeing its possible superstrings. On the other hand, domain experts often encounter the case where every string in a focused equivalence class is a fragment of a word (or phrase) and seems to be meaningless. In this case, they need to see other equivalence classes whose members are superstrings of the representative of the focused equivalence class. Such equivalence classes will be encountered elsewhere in the list. Domain experts do not necessarily work through the list sequentially when examining equivalence classes.

5.2. Superstring–substring relation on strings

As described above, it is necessary to provide domain experts with a tool for browsing the superstring–substring relation among the representatives. One reasonable approach would be to draw a Hasse diagram representing this partial order, or to draw the symmetric compact DAWG for S described in the previous section. Here, we briefly discuss the relationship between the two graph structures.

Definition 9. For any x, y in Σ^*, let $x \preccurlyeq y$ if and only if x is a substring of y. We write $x \prec y$ if $x \preccurlyeq y$ and $x \neq y$.

Definition 10. For any x, y in $Prime(S)$, let $x \lhd y$ if and only if $x \prec y$, and there is no element z in $Prime(S)$ such that $x \prec z \prec y$.

The Hasse diagram is a well-known structure for depicting a finite, partially ordered set. The Hasse diagram for $\langle Prime(S), \preccurlyeq \rangle$ is the directed acyclic graph $H(S) = (V, E)$, where $V = Prime(S)$ is the set of vertices, and $E = \{(x, y) \in V \times V \mid x \lhd y\}$ is the set of arcs.

Lemma 4. *For arbitrary strings x and y in $Prime(S)$, $x \triangleleft y$ implies that the symmetric compact DAWG for S has an arc from node x to node y.*

Proof. It can be seen that, for arbitrary strings x, y in $Prime(S)$, $x \triangleleft y$ holds only if there exists a symbol σ in Σ such that $\overleftrightarrow{\sigma x} = y$ or $\overleftrightarrow{x\sigma} = y$. □

Note that the inverse implication does not hold in general. It can be seen in Fig. 6 that there is an arc from node b to node $babb$, but $b \triangleleft babb$ does not hold because of $b \prec ab \prec babb$. In this sense, the symmetric compact DAWG for S gives us much more information than the Hasse diagram $H(S)$. That is, the labeled edges of the symmetric compact DAWG represent: *What happens when appending a possible character to the left or right end of a prime substring*?

5.3. Browsing the relation

As above, the symmetric compact DAWG is more informative than the Hasse diagram. However, both the graphs for real data are too large to draw. A reasonable solution to such cases is to draw only the "neighbor" of a user-specified node. The user can traverse the whole graph by repeatedly shifting his or her focus.

Suppose the user is now interested in a node $x \in Prime(S)$. We define the *superstring graph* of x to be the subgraph of the symmetric compact DAWG for S consisting only of the nodes which are reachable from the node x, and the *substring graph* of x to be the subgraph of the symmetric compact DAWG for S consisting only of the nodes from which the node x is reachable. When the string x is relatively short, the substring graph of x is of reasonable size. However, the superstring graph of x is still large and complicated. See, for example, Fig. 7. This is a subgraph of the symmetric compact DAWG for Kokin-Shū and Gosen-Shū, and it consists only of the nodes reachable from the node "KA-SU-MI". That is, it is the superstring graph of "KA-SU-MI".

For this reason, we decided to use other data structures, which we call *context trees*, instead of the superstring graph. By repeated uses of the context trees we can obtain the same information as supplied by the superstring graph.

The *right context tree* of a prime substring x is essentially the same as the subtree of the node x in the suffix tree for S, but augmented by adding to every node xy, $xy = \overrightarrow{xy}$ ($y \in \Sigma^*$), a label "$\gamma[x]y$" such that $\gamma xy = \overleftrightarrow{xy}$. The left context tree is defined symmetrically. There may be two nodes xy_1 and xy_2 such that $y_1 \neq y_2$, but $\overleftrightarrow{xy_1} = \overleftrightarrow{xy_2}$. Therefore, more than one node may have the same label if ignoring the square brackets ([,]). Fig. 8 shows the left and the right context trees of string b for $S = \{babbc, ababb\}$.

Fig. 9 shows the context trees of "KA-SU-MI" for Kokin-Shū and Gosen-Shū. Compare the context trees in Fig. 9 with the superstring graph in Fig. 7. The right context tree of "KA-SU-MI" is composed of the nodes that are reachable from it only via the arcs in E_R (the solid arrows). Conversely, the left context tree is composed of the nodes that are reachable only via the arcs in E_L (the broken arrows). The superstring graph of "KA-SU-MI" contains nodes that are not in its context trees. Such nodes, however, can be reached by repeatedly changing the focus of the context trees.

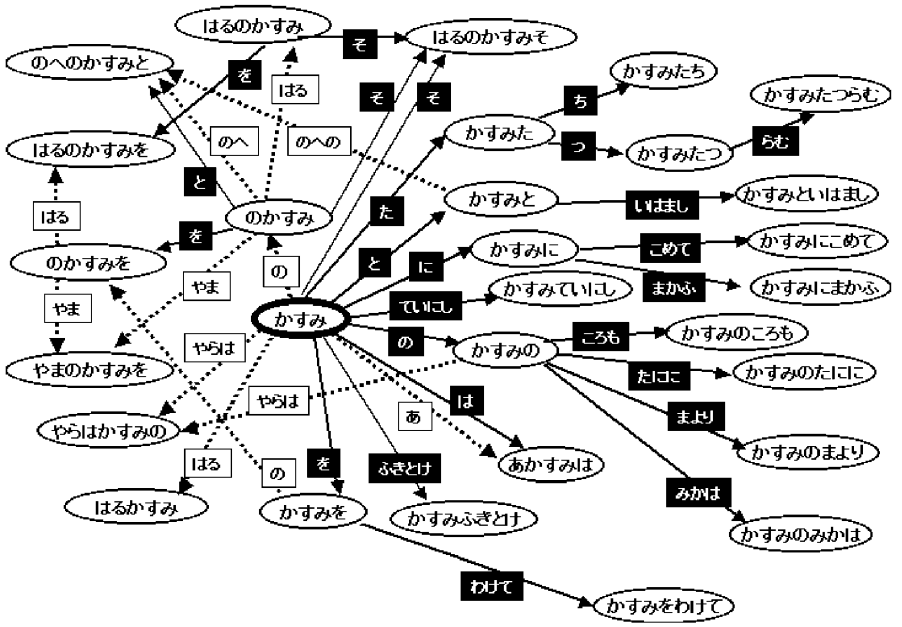

Fig. 7. The superstring graph of string "KA-SU-MI" for Kokin-Shū and Gosen-Shū. The ellipses represent the nodes, and the solid and the broken arrows the labeled arcs in E_R and in E_L, respectively. The strings in rectangles represent the edge labels. The ellipse emphasized by the thick line is the node labeled "KA-SU-MI".

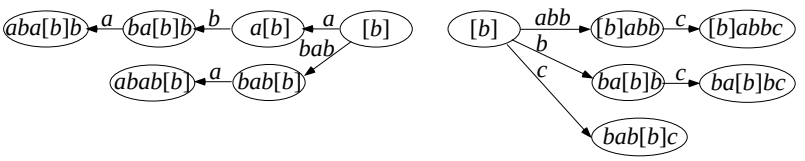

Fig. 8. Context trees of b for $S = \{babbc, ababb\}$. The left tree is the left context tree of b, and the right tree is the right context tree of b. Note that the nodes are labeled by strings of the form $x[b]y$ $(x, y \in \Sigma^*)$. The right (resp. left) context tree of string b is essentially the same as the subtree of the node labeled b in the suffix tree (resp. prefix tree) for S, if we substitute by (resp. xb) for label strings $x[b]y$.

The context trees are useful for grasping the possible superstrings of one focused string at a time. This is a desirable feature for domain experts who must access the context of each appearance of a particular expression in a large corpus. Most of the existing text analysis tools have the key word in context (KWIC) display [10]. The context trees enable researchers to grasp the context of a string more quickly compared with KWIC, especially when the string occurs frequently.

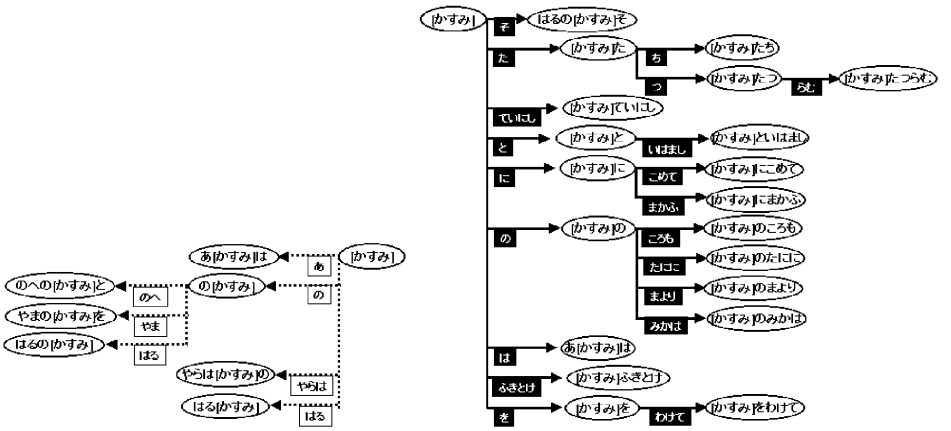

Fig. 9. Context trees of "KA-SU-MI" for Kokin-Shū and Gosen-Shū. The left tree is the left context tree of "KA-SU-MI" and the right tree is the right context tree of the same string.

5.4. Compact representation of equivalence classes

As above, every member in a non-degenerate equivalence class under $\equiv$ is a substring of the representative of the class. There can be $O(m^2)$ members in a non-degenerate equivalence class, where m is the length of the representative of the class. Hence, we need a compact representation of an equivalence class when the representative is relatively long. (By "representation" we mean a means of illustrating the members of an equivalence class, not a data structure inside the computer.)

Lemma 5. *Let $x \in Prime(S)$ for a set S of strings. Then, every $y \in [x]_{\equiv}$ occurs within the string x once and only once.*

Lemma 6. *Let $x \in Prime(S)$ for a set S of strings. Let $y \in [x]_{\equiv}$. Then, every string z such that $y \preccurlyeq z \preccurlyeq x$ is a member of $[x]_{\equiv}$.*

Since the representative of a non-degenerate equivalence class is a unique maximal (maximum) element with respect to the partial order $\preccurlyeq$, we have the following lemma.

Lemma 7. *Let $x \in Prime(S)$ for a set S of strings. Let $y_1, \ldots, y_k$ be the minimal elements of $[x]_{\equiv}$. Then,*

$$[x]_{\equiv} = Pincer(y_1, x) \cup \cdots \cup Pincer(y_k, x),$$

where $Pincer(y_i, x)$ is the set of strings z with $y \preccurlyeq z \preccurlyeq x$.

For example, see Table 2 again. The class ranked 188th has five members, and the strings "MA-TA-RE-KE" and "YA-MA-TA-RE" are both minimal. On the other hand, the class ranked 200th has eight members, and only the string "HI-YA-WA" is minimal.

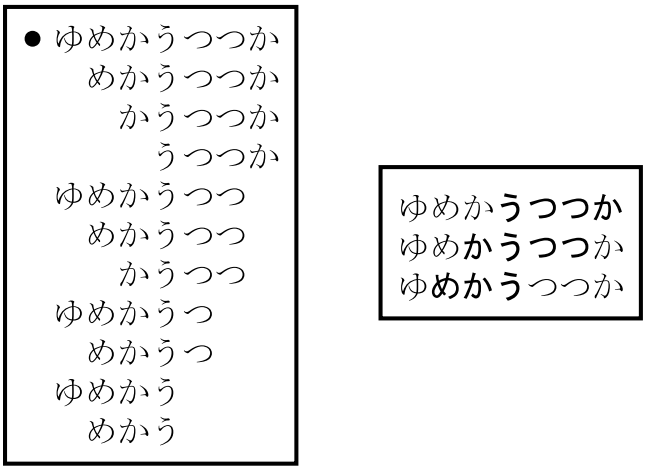

Fig. 10. A compact representation of an equivalence class. The left box is an equivalence class for Kokin-Shū and Gosen-Shū, where ● indicates the representative. The right box is a compact representation of it, in which the bold-faced substrings of the representative are the minimal members of the class.

Concerning the number of minimal elements, we have the following results.

Lemma 8. *Let $x \in Prime(S)$ for a set S of strings. Then, there are at most $|x|$ minimal elements in $[x]_{\equiv}$.*

Proof. We can represent a minimal element y of $[x]_{\equiv}$ as a pair (s,t) of integers, with $1 \leqslant s \leqslant t \leqslant |x|$, such that $y = x[s : t]$. By Lemma 5 such integers are uniquely determined. Suppose that the minimal elements are represented by $(s_1,t_1),\ldots,(s_k,t_k)$. These intervals may have overlaps, but they cannot subsume each other. Hence, the integers $s_1,\ldots,s_k$ must be distinct. (The integers $t_1,\ldots,t_k$ must also be distinct.) The proof is now complete. □

We thus have the following results.

Theorem 1. *A non-degenerate equivalence class under $\equiv$ can be represented by its representative x, and at most $|x|$ substrings of x.*

Fig. 10 shows a compact representation of an equivalence class based on this idea.

6. Experimental results

6.1. Comparison of two imperial anthologies

First we compared Kokin-Shū and Shin-Kokin-Shū, which are known as the best two of the 21 imperial anthologies, and have been studied extensively. Kokin-Shū was

compiled in 922 and Shin-Kokin-Shū in 1205. Some of the expressions that differentiate the two anthologies may be a reflection of literary trends, and we expected that we could give a characterization of each anthology from the viewpoint of expressions.

We created a list of equivalence classes, and examined them. There were a few expressions which are reflections of the time difference. For example, Kokin-Shū has 15 examples of "HE-RA-NA-RI", ranked 30th, but there are none in Shin-Kokin-Shū. This is a form of auxiliary verb that is known to have been only used in the period of Kokin-Shū. It is, however, a well-known fact and is therefore far from being a discovery. It seems less interesting to enumerate differences between the two anthologies because there are apparently many differences between them. We cannot deny the possibility that potentially there may exist some expressions which differentiate the anthologies' characteristics in a non-trivial manner, but it is not easy to establish such a new body of theory on this issue.

6.2. Comparison of two private anthologies

We considered a comparison of two anthologies which necessarily resemble each other. As pairs of such anthologies, we chose the following two pairs of private anthologies:

(A) Sanka-Shū by the priest Saigyō (1118–1190), and Shūgyoku-Shū by the priest Jien (1155–1225). It is well-known that Saigyō had a great influence on Jien. In fact, Jien composed many poems using similar expressions to those preferred by Saigyō.

(B) Shūi-Gusō by Fujiwara-no-Teika (1162–1241), and Tameie-Shū by Fujiwara-no-Tameie (1198–1275). Teika was a poet and a literary theorist, who ranks among the greatest of the Waka poets. The poems of his son Tameie were influenced by the poems and the theory of Teika.

Table 5 shows two lists of equivalence classes for: (A) Sanka-Shū and Shūgyoku-Shū, and (B) Shūi-Gusō and Tameie-Shū. We used information entropy as the statistical measure.

From the upper part of the list, we note the following by using a prototype of the browser for context trees:

- While Shūgyoku-Shū has Buddhist terms such as "Nori-no-Michi" (the road of dharma), "Nori-no-Hana" (the flower of dharma), and "Makoto-no-Michi" (the road of truth) in 20, 16, and 18 poems, respectively, there is no such expression in Sanka-Shū. It should be noted that the first two terms were obtained by browsing the right context tree of the string "NO-RI-NO", which ranked 15th, and the last was obtained via the left context tree of the string "NO-MI-CHI", which ranked 24th. See Fig. 11.
- There are 21 examples of "···wo Ikanisemu" (most of them are "Mi wo Ikanisemu") in Shūgyoku-Shū and there are none in Sanka-Shū. This is interesting and seems to suggest differences in their belief in Buddhism and in their way of life. The expression "wo Ikanisemu" was obtained via the left context tree of the string "I-KA-NI-SE", which ranked 31st.
- In Tameie-Shū, there are many expressions using "Oi-no-Nezame" (wakeful night for the aged) and "Oi-no-Namida" (tears of the aged), but there are no such expressions

Table 5
Lists of equivalence classes for two pairs of private anthologies

	P_1	N_1	Equivalence class
(A) Sanka-Shū vs. Shūgyoku-Shū			
1	33	397	• RU-NO
2	26	351	• HA-RU-NO
3	16	271	• MI-YO
4	919	2822	• TE
5	29	344	• NO-SO
6	41	28	• KO-KO-CHI
7	44	33	• KO-CHI
8	21	273	• NO-SO-RA
9	60	495	• SO-RA
10	7	163	• MI-YO-SHI
11	3	120	• SU-MI-YO
12	99	167	• MA-SHI
13	7	150	• MI-YO-SHI-NO
14	3	114	• SU-MI-YO-SHI
15	7	147	• NO-RI-NO
16	487	2281	• YO
17	53	418	• YU-FU
18	8	150	• TSU-KA-SE
19	0	67	• NO-KO-RU
20	117	722	• HA-RU
21	1354	5327	• NO
22	3	101	• SU-MI-YO-SHI-NO
23	2	90	• YO-HA-NO
24	0	62	• NO-MI-CHI
25	14	3	• KA-NA-SHI-KA
26	1	75	• KO-RU
27	9	0	• TSU-TSU-MA
28	55	79	• NI-TE
29	6	119	• MA-TSU-KA-SE
30	1	73	• SU-MI-NO
31	4	102	• I-KA-NI-SE
32	37	306	• YA-MA-NO
33	152	848	• KA-SE
34	2	81	• HO-TO-KE
35	129	744	• A-KI
36	330	912	• TSU-KI
37	23	223	• NA-HO
38	21	211	• NO-RI
39	19	11	• MI-KE
40	74	131	• TE-NI
(B) Shūi-Gusō vs. Tameie-Shū			
1	14	103	• O-I
2	2	63	• O-I-NO
3	6	54	• WO-KU-RA
4	11	60	• WO-KU
5	246	74	• SO-RA
6	4	38	• NI-KE-RU

Table 5 (*continued*)

	P_1	N_1	Equivalence class
7	109	168	• KO-SO
8	3	34	• KU-RA-YA-MA
9	54	106	• I-NO
10	3	33	• WO-KU-RA-YA-MA WO-KU-RA-YA
11	0	23	• O-I-NO-NE I-NO-NE
12	69	7	• NA-KA-ME
13	6	35	• KU-RA-YA
14	0	19	• O-I-NO-NE-SA-ME I-NO-NE-SA-ME O-I-NO-NE-SA I-NO-NE-SA
15	88	16	• SO-TE-NO
16	292	115	• I-RO
17	57	99	• NI-KE
18	53	6	• KI-E
19	9	36	• RA-YA-MA
20	2	22	• RA-NO-YA-MA
21	77	114	• NA-MI-TA
22	53	7	• KU-YO
23	2	21	• WO-KU-RA-NO
24	141	173	• KA-MI
25	234	92	• SO-TE
26	39	3	• I-KU-YO
27	134	166	• NA-RI
28	42	74	• KE-RE
29	54	8	• HO-HI
30	2	20	• NO-NE-SA-ME
31	91	123	• RI-KE
32	47	6	• NI-HO-HI
33	0	13	• WO-KU-RA-NO-YA-MA-NO
34	4	23	• RA-NO-YA
35	208	226	• NA-SHI

in Shūi-Gusō. Although Tameie was conscious of old age in his poems, he did not live longer than his father. (Teika died at 80 and Tameie at 78.) Surveying Shinpen-Kokkataikan, a collection of 1,162 anthologies of Waka poems (about 450 000 poems in total), we find that the expressions "Oi-no-Nezame" and "Oi-no-Namida" most frequently appear in Tameie-Shū. These expressions, therefore, definitely characterize Tameie's poetry. In his last days, he was involved in a family feud about his successor. It resulted in dividing the dynasty into three as Nijō, Kyōgoku, and Reizei in the next generation. This is quite a contrast to the case of Teika, who could decide Tameie was his only successor. Note that these characteristic expressions were obtained via the right context tree of "O-I", which ranked as the best.

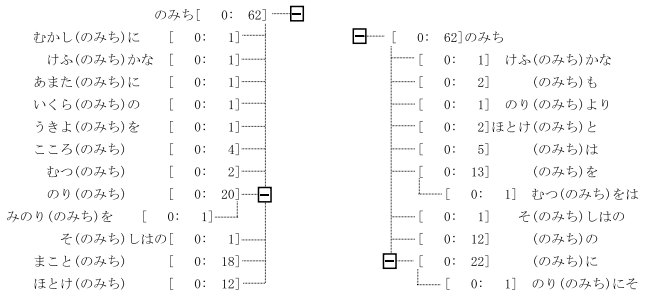

Fig. 11. The left and right trees are, respectively, the left and the right context trees of "NO-MI-CHI". In both trees, the top string is "NO-MI-CHI". In the left context tree, the 9th and 12th strings from the top are "NO-RI-NO-MI-CHI" and "MA-KO-TO-NO-MI-CHI", respectively.

7. Concluding remarks

In this paper, we addressed the problem of discovering characteristic expressions from literary works. We believe that a key to success is a method to support domain experts involved in the interpretation/evaluation of mined patterns, and we have presented two ideas to support them. One is to partition text substrings into equivalence classes under an equivalence relation on strings, and to create a list of these and let the experts examine the equivalence classes one by one. The other is to draw left and right context trees of a focused string, so that the experts can see its superstrings as well as its contexts. These ideas were effective and we succeeded in finding characteristic expressions from two pairs of private anthologies, which are stimulating and provide Waka researchers with clues for further investigation.

It is also desirable to establish a means of extracting characteristic expressions from prose texts. We have applied the same method to prose texts, but most of the strings collected from these are proper nouns such as characters' names, titles and place names. These largely depend on the story settings and the characters. Finding an effective filtering technique for removing such strings is planned for future work.

References

[1] H. Arimura, Text data mining with optimized pattern discovery, in: Proc. 17th Workshop on Machine Intelligence, Cambridge, 2000.

[2] A. Blumer, J. Blumer, D. Haussler, R. Mcconnell, A. Ehrenfeucht, Complete inverted files for efficient text retrieval and analysis, J. ACM 34(3) (1987) 578–595, previous version in: STOC'84.

[3] M. Crochemore, W. Rytter, Text Algorithms, Oxford University Press, Oxford, 1994.

[4] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, Berlin, 1997.

[5] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery: an overview, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press/The MIT Press, Cambridge, MA, 1996, pp. 1–34.

[6] T. Fukuda, Tametada-shū revisited, in: Proc. 46th Symp. on Waka Literature, 2000.

[7] T. Fukuda, "as if it were in darkness/my parental heart/is blind and lost in …"—Kanesuke's poem revisited, 2001, submitted for publication.

[8] Z. Galil, Open problems in stringology, in: A. Apostolico, Z. Galil (Eds.), Combinatorial Algorithms on Words, NATO ASI Series, NATO Advanced Science Institutes Series, Series F: Computer and Systems Sciences, Vol. 12, Springer, Berlin, 1985, pp. 1–8.
[9] M. Kondo, Studies on classical Japanese literature based on string analysis using *n*-gram statistics, Technical Report, Chiba University, March 2000 (in Japanese).
[10] H. Luhn, Keyword-in-context index for technical literature (KWIC index), Am. Doc. 11 (1960) 288–295.
[11] S. Morishita, On classification and regression, in: Proc. First Int. Conf. on Discovery Science, Lecture Notes in Artificial Intelligence, Vol. 1532, 1998, pp. 49–59.
[12] M. Murakami, Y. Imanishi, On a quantitative analysis of auxiliary verbs used in Genji Monogatari, Trans. Process. Soc. Jpn. 40 (3) (1999) 774–782 (in Japanese).
[13] S. Shimozono, H. Arimura, S. Arikawa, Efficient discovery of optimal word-association patterns in large databases, New Gener. Comput. 18 (1) (2000) 49–60.
[14] M. Takeda, T. Fukuda, I. Nanri, M. Yamasaki, K. Tamari, Discovering instances of poetic allusion from anthologies of classical Japanese poems, Theoret. Comput. Sci. (this Vol.) (2003).